Raj Kumar

Professor Smallberg

Computer Science 32

2 March 2021

Homework 4

**Question 2: Explain in a sentence or two why the call to the one-argument form of Sequence<Coord>::insert causes at least one compilation error. (Notice that the call to the one-argument form of Sequence<int>::insert is fine, as is the call to the two-argument form of Sequence<Coord>::insert.) Don't just transcribe a compiler error message; your answer must indicate you understand the ultimate root cause of the problem and why that is connected to the call to Sequence<Coord>::insert.**

The one argument form causes an error because the insert function uses the < operator to compare two objects of ItemType, which in this case is a Coord. The < operator (or <=, >, >=) isn’t defined for two Coords, so the compiler doesn’t what to do and communicates this with an error.

**Question 4b: We introduced the two-parameter overload of listAll. Why could you not solve this problem given the constraints in part a if we had only a one-parameter listAll, and you had to implement *it* as the recursive function?**

This would be impossible because then the recursive calls of the listAll function would not be aware of the path that the file had come from, as there would be no way of sharing this information between where the function was called and the instance of the function that was actually created. This is because the constraints only allow a single variable, which is a pointer to the file itself, and do not allow other ways of sharing the path for the file.

**Question 5a: What is the time complexity of this algorithm, in terms of the number of basic operations (e.g., additions, assignments, comparisons) performed: Is it O(N), O(N log N), or what? Why?**

O(N^3), because everything other than the loop within the loop within the loop (all which would iterate up to N times) would have constant time complexity. Thus, the time complexity would simplify to O(N^3) because N\*N\*N = N^3.

**Question 5a: What is the time complexity of this algorithm? Why?**

This is also O(N^3) because, like part a, everything other than the loop within the loop within the loop would have constant time complexity, and the middle loop (the loop within the outermost loop) would execute I times, and I varies from 0 to N - 1. Thus, on average, the middle loop with I would run N/2 times, and since constant multiples do not matter, this would also have O(N^3) complexity as the outermost loop and the most innermost loop both execute up to N times, and N\*N\*N = N^3.

**Question 6a: Assume that seq1, seq2, and the old value of result each have N elements. In terms of the number of ItemType objects visited (in the linked list nodes) during the execution of this function, what is its time complexity? Why?**

The time complexity would be O(N^2) in terms of ItemType objects visited, because inside both loops, which both execute the loop bodies N times, multiple functions with O(N) time complexity like the get function and the insert function are present. The insert function with an ItemType input and an integer input has O(N) complexity in terms of how many ItemType objects or pointers are visited because the function calls both nodeAtPos and insertBefore functions, and the nodeAtPos has to traverse the list of N elements up to N times, and the insertBefore function is constant time complexity because is no iteration and there is only constant time complexity actions. This means that this overload of the insert function has time complexity O(N). The get function is also O(N) time complexity in terms of ItemType objects visited because the list has to go through the list of N elements up to N times. Thus, since both loops in the interleave function have O(N) function calls, and both loops run N times, the overall time complexity for interleave becomes O(N^2), because N\*N = N^2.

Note: the swap function just switches pointers, which is O(1) time complexity.

**Question 6b: Assume that**seq1**,**seq2**, and the old value of**\*this**each have about N elements. In terms of the number of ItemType objects visited during the execution of this function, what is its time complexity? Why? Is it the same, better, or worse, than the implementation in part a?**

This implementation has two loops which iterate up to N times, and both include the insertBefore function and no other functions or loops, which means everything else has constant time complexity. The insertBefore function is O(1) time complexity because is no iteration and there is only constant time complexity actions. Thus, the overall time complexity of this interleave function is O(N), because N\*1=N. Thus, the interleave implementation in part b is better than the implementation in part a, as the implementation in part b has a lower time complexity growth factor.

Note: the swap function just switches pointers, which is O(1) time complexity.